XML

From Wikipedia, the free encyclopedia

Extensible Markup Language	
File extension:	`.xml`
MIME type:	`application/xml`, `text/xml` (deprecated)
Uniform Type Identifier:	public.xml
Developed by:	World Wide Web Consortium
Type of format:	Markup language
Extended from:	SGML
Extended to:	XHTML, RSS, Atom, ...
Standard(s):	1.0 (Fourth Edition) (http://www.w3.org/TR/2006/REC-xml-20060816/) 1.1 (Second Edition) (http://www.w3.org/TR/2006/REC-xml11-20060816/)

The **Extensible Markup Language** (**XML**) is a general-purpose markup language.[1] It is classified as an extensible language because it allows its users to define their own tags. Its primary purpose is to facilitate the sharing of data across different information systems, particularly via the Internet.[2]

It is a simplified subset of the Standard Generalized Markup Language (SGML), and is designed to be relatively human-legible. By adding semantic constraints, application languages can be implemented in XML. These include XHTML,[3] RSS, MathML, GraphML, Scalable Vector Graphics, MusicXML, and thousands of others. Moreover, XML is sometimes used as the specification language for such application languages.

XML is recommended by the World Wide Web Consortium. It is a fee-free open standard. The W3C recommendation specifies both the lexical grammar, and the requirements for parsing.

Contents

- 1 Well-formed and valid XML documents
- 2 Well-formed documents: XML syntax
  - 2.1 Entity references
    - 2.1.1 Numeric character references
  - 2.2 Well-formed documents
  - 2.3 Automatic verification
- 3 Valid documents: XML semantics
  - 3.1 DTD
  - 3.2 XML Schema
  - 3.3 RELAX NG
  - 3.4 ISO DSDL and Other Schema Languages
  - 3.5 International use
- 4 Displaying XML on the web
- 5 XML extensions
- 6 Processing XML files
- 7 History
  - 7.1 Sources
  - 7.2 Versions
  - 7.3 Patent claims
- 8 Critique of XML
  - 8.1 Advantages of XML

- 8.2 Disadvantages of XML
- 9 See also
- 10 Notes and references
- 11 External links
  - 11.1 Specifications
  - 11.2 Parsers
  - 11.3 Sources
  - 11.4 Retrospectives
  - 11.5 Papers

Well-formed and valid XML documents

There are two levels of correctness of an XML document:

- **Well-formed**. A well-formed document conforms to all of XML's syntax rules. For example, if an element has an opening tag with no closing tag and is not self-closing, it is not *well-formed*. A document that is not well-formed is not considered to be XML; a *conforming parser* is not allowed to process it.
- **Valid**. A valid document additionally conforms to some semantic rules. These rules are either user-defined, or included as an XML schema or DTD. For example, if a document contains an undefined tag, then it is not *valid*; a *validating parser* is not allowed to process it.

Well-formed documents: XML syntax

As long as only well-formedness is required, XML is a generic framework for storing any amount of text or any data whose structure can be represented as a tree structure. The only indispensable syntactical requirement is that the document have exactly one **root element** (alternatively called the **document element**). This means that the text must be enclosed between a root opening tag and a corresponding closing tag. The following is a well-formed XML document:

```
<book>Gallia omnia divisa est in partes tres .... </book>
```

The root element can be preceded by an optional **XML declaration**. This element states what version of XML is in use (normally 1.0); it may also contain information about character encoding and external dependencies.

```
<?xml version="1.0" encoding="UTF-8"?>
```

The specification *requires* that processors of XML support the pan-Unicode character encodings UTF-8 and UTF-16 (UTF-32 is not mandatory). The use of more limited encodings, such as those based on ISO/IEC 8859, is acknowledged and is indeed widely used and supported.

Comments can be placed anywhere in the tree, including in the text if the content of the element is text or #PCDATA:

```
<!-- This is a comment. -->
```

In any meaningful application, additional markup is used to structure the contents of the XML document. The text enclosed by the root tags may contain an arbitrary number of XML elements. The basic syntax for one **element** is:

```
<name attribute="value">content</name>
```

Here, »content« is some text which may again contain XML elements. So, a generic XML document contains a tree-based data structure. In this respect, it is similar to the LISP programming language's S-expressions, which describe tree structures wherein each node may have its own property list. Here is an example of a structured XML document:

```
<book title="Mécanique" authors="Landau, Lifshitz">
  <chapter title="équations du mouvement">
     <section title="coordonnées généralisées">
        Une des notions fondamentales de la Mécanique est celle de point matériel. ....
     </section>
     <section title="Le principe de moindre action">
        La formule la plus générale de la loi du mouvement des systèmes mécaniques est
        fournie par le principe dit de moindre action (ou principe de Hamilton). ....
     </section>
     ....
   </chapter>
   <chapter title = "lois de conservation">
     ...
   </chapter>
   ...
</book>
```

Attribute values must always be quoted, using single or double quotes, and each attribute name should appear only once in any element. XML requires that elements be properly nested — elements may never overlap. For example, the code below is not well-formed XML, because the *em* and *strong* elements overlap:

```
<!-- WRONG! NOT WELL-FORMED XML! -->
<p>Normal <em>emphasized <strong>strong emphasized</em> strong</strong></p>
```

XML provides special syntax for representing an element with empty content. Instead of writing a start tag followed immediately by an end tag, a document may contain an empty-element tag. An empty-element tag resembles a start tag but contains a slash just before the closing angle bracket. The following three examples are equivalent in XML:

```
<foo></foo>
<foo />
<foo/>
```

An empty-element tag may contain attributes:

```
<info author="John" genre="science-fiction" date="2009-Jan-01" />
```

Entity references

XML provides two methods for referring to special characters: character entity references and numeric character references.

An entity in XML is a named body of data, usually text, such as an unusual character.

An entity reference is a placeholder that represents that entity. It consists of the entity's name preceded by an ampersand ("&") and followed by a semicolon (";"). XML has five predeclared entities:

- & (&, ampersand)
- < (<, less than)
- > (>, greater than)
- ' (', apostrophe)
- " (", quotation mark)

Here is an example using a predeclared XML entity to represent the ampersand in the name "AT&T":

```
<company-name>AT&amp;T</company-name>
```

If more entities need to be declared, this is done in the document's Document Type Definition (DTD). A basic example of doing so in a minimal internal DTD follows. Declared entities can describe single characters or pieces of text, and can reference each other.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE example [
        <!ENTITY copy "&#xA9;">
        <!ENTITY copyright-notice "Copyright &copy; 2006, XYZ Enterprises">
]>
<example>
        &copyright-notice;
</example>
```

When viewed in a suitable browser, the XML document above appears as:

```
<example> Copyright © 2006, XYZ Enterprises </example>
```

Numeric character references

Numeric character references look like entity references, but instead of a name, they contain the "#" character followed by a number. The number (in decimal or "x"-prefixed hexadecimal) represents a Unicode code point. Unlike entity references, they are neither predeclared nor do they need to be declared in the document's DTD. They have typically been used to represent characters that are not easily encodable, such as an Arabic character in a document produced on a European computer. The ampersand in the "AT&T" example could also be escaped like this (decimal 38 and hexadecimal 26 both

represent the Unicode code point for the "&" character):

```
<company_name>AT&#38;T</company_name>
<company_name>AT&#x26;T</company_name>
```

There are many more rules necessary to be sure of writing well-formed XML documents, such as the use of namespaces and the exact characters allowed in an XML name, but this quick tour provides the basics necessary to read and understand many XML documents.

Well-formed documents

contents of this subsection should be absorbed above

A well-formed document must conform to the following rules, among others:

- Non-empty elements are delimited by both a start-tag and an end-tag.
- Empty elements may be marked with an empty-element (self-closing) tag, such as `<IAmEmpty />`. This is equal to `<IAmEmpty></IAmEmpty>`.
- All attribute values are quoted with either single (') or double (") quotes. Single quotes close a single quote and double quotes close a double quote.
- Tags may be nested but must not overlap. Each non-root element must be completely contained in another element.
- The document complies with its declared character encoding. The encoding may be declared or implied externally, such as in "Content-Type" headers when a document is transported via HTTP, or internally, using explicit markup at the very beginning of the document. When no such declaration exists, a Unicode encoding is assumed, as defined by a Unicode Byte Order Mark before the document's first character. If the mark does not exist, UTF-8 encoding is assumed.

Element names are case-sensitive. For example, the following is a well-formed matching pair:

```
<Step> ... </Step>
```

whereas this is not

```
<Step> ... </step>
```

The careful choice of names for XML elements will convey the meaning of the data in the markup. This increases human readability while retaining the rigor needed for software parsing.

Choosing meaningful names implies the semantics of elements and attributes to a human reader without reference to external documentation. However, this can lead to verbosity, which complicates authoring and increases file size.

Automatic verification

It is relatively simple to verify that a document is well-formed XML:

- load it into a browser

- use a tool like xmlwf (usually bundled with expat)
- parse the document, for instance in Ruby:

```
irb> require "rexml/document"
irb> include REXML
irb> test = Document.new(File.new("test.xml")).root
```

Valid documents: XML semantics

By leaving the names, allowable hierarchy, and meanings of the elements and attributes open and definable by a customizable *schema or DTD*, XML provides a syntactic foundation for the creation of purpose specific, XML-based markup languages. The general syntax of such languages is rigid — documents must adhere to the general rules of XML, assuring that all XML-aware software can at least read and understand the relative arrangement of information within them. The schema merely supplements the syntax rules with a set of constraints. Schemas typically restrict element and attribute names and their allowable containment hierarchies, such as only allowing an element named 'birthday' to contain 1 element named 'month' and 1 element named 'day', each of which has to contain only character data. The constraints in a schema may also include data type assignments that affect how information is processed; for example, the 'month' element's character data may be defined as being a month according to a particular schema language's conventions, perhaps meaning that it must not only be formatted a certain way, but also must not be processed as if it were some other type of data.

An XML document that complies with a particular schema/DTD, in addition to being well-formed, is said to be **valid**.

An XML schema is a description of a type of XML document, typically expressed in terms of constraints on the structure and content of documents of that type, above and beyond the basic constraints imposed by XML itself. A number of standard and proprietary XML schema languages have emerged for the purpose of formally expressing such schemas, and some of these languages are XML-based, themselves.

Before the advent of generalised data description languages such as SGML and XML, software designers had to define special file formats or small languages to share data between programs. This required writing detailed specifications and special-purpose parsers and writers.

XML's regular structure and strict parsing rules allow software designers to leave parsing to standard tools, and since XML provides a general, data model-oriented framework for the development of application-specific languages, software designers need only concentrate on the development of rules for their data, at relatively high levels of abstraction.

Well-tested tools exist to validate an XML document "against" a schema: the tool automatically verifies whether the document conforms to constraints expressed in the schema. Some of these validation tools are included in XML parsers, and some are packaged separately.

Other usages of schemas exist: XML editors, for instance, can use schemas to support the editing process (by suggesting valid elements and attributes names, etc).

DTD

The oldest schema format for XML is the Document Type Definition (DTD), inherited from SGML. While DTD support is ubiquitous due to its inclusion in the XML 1.0 standard, it is seen as limited for the following reasons:

- It has no support for newer features of XML, most importantly namespaces.
- It lacks expressivity. Certain formal aspects of an XML document cannot be captured in a DTD.
- It uses a custom non-XML syntax, inherited from SGML, to describe the schema.

XML Schema

A newer XML schema language, described by the W3C as the successor of DTDs, is XML Schema, or more informally referred to by the initialism for XML Schema instances, XSD (XML Schema Definition). XSDs are far more powerful than DTDs in describing XML languages. They use a rich datatyping system, allow for more detailed constraints on an XML document's logical structure, and must be processed in a more robust validation framework. XSDs also use an XML-based format which makes it possible to use ordinary XML tools to help process them, although XSD implementations require much more than just the ability to read XML.

Criticisms of XSD include the following:

- The specification is very large, which makes it difficult to understand and implement.
- The XML-based syntax leads to verbosity in schema description, which makes XSDs harder to read and write.
- Schema validation can be an expensive addition to XML parsing, especially for high volume systems.
- The modeling capabilities are very limited, with no ability to allow attributes to influence content models.
- The type derivation model is very limited, in particular that derivation by extension is rarely useful.
- Database-related data transfer has been supported with arcane ideas such as nillability but the requirements of industrial publishing are under-supported.
- The key/keyref/uniqueness mechanisms are not type aware
- The PSVI concept (Post Schema Validation Infoset) does not have a standard XML representation or API, thus it works against vendor independence unless revalidation is performed.

RELAX NG

Another popular schema language for XML is RELAX NG. Initially specified by OASIS, RELAX NG is now also an ISO international standard (as part of DSDL). It has two formats: an XML based syntax and a non-XML compact syntax. The compact syntax aims to increase readability and writability but, since there is a well-defined way to translate the compact syntax to the XML syntax and back again by means of James Clark's Trang conversion tool (http://www.thaiopensource.com/relaxng/trang.html), the advantage of using standard XML tools is not lost. RELAX NG has a simpler definition and validation framework than XML Schema, making it easier to use and implement. It also has the ability to use datatype framework plug-ins; a RELAX NG schema author, for example, can require values in an XML document to conform to definitions in XML Schema Datatypes.

ISO DSDL and Other Schema Languages

The ISO DSDL (Document Schema Description Languages) standard brings together a comprehensive set of small schema languages, each targeted at specific problems. DSDL includes RELAX NG full and compact syntax, Schematron assertion language, and languages for defining datatypes, character repertoire constraints, renaming and entity expansion, and namespace-based routing of document fragments to different validators. DSDL schema languages do not have the vendor support of XML Schemas yet, and are to some extent a grassroots reaction of industrial publishers to the lack of utility of XML Schemas for publishing.

Some schema languages not only describe the structure of a particular XML format but also offer limited facilities to influence processing of individual XML files that conform to this format. DTDs and XSDs both have this ability; they can for instance provide attribute defaults. RELAX NG and Schematron intentionally do not provide these infoset augmentation facility.

International use

XML supports the direct use of almost any Unicode character in element names, attributes, comments, character data, and processing instructions. Therefore, the following is a well-formed XML document, even though it includes both Chinese and Cyrillic characters:

```
<?xml version="1.0" encoding="UTF-8"?>
<俄语>Данные</俄语>
```

Displaying XML on the web

Because XML is a *data description* language, XML documents do not carry information about how to display the data. Without using CSS or XSL, a generic XML document is rendered as raw XML text by most web browsers. Some display it with 'handles' (e.g. + and - signs in the margin) that allow parts of the structure to be expanded or collapsed with mouse-clicks.

In order to style the rendering in a browser with CSS, the XML document must include a reference to the stylesheet:

```
<?xml-stylesheet type="text/css" href="myStyleSheet.css"?>
```

See the CSS article for an example of this in action.

Note that this is different from specifying such a stylesheet in HTML, which uses the <link> element.

Extensible Stylesheet Language (XSL) can be used to alter the format of XML data, either into HTML or other formats that are suitable for a browser to display.

To specify client-side XSL Transformation (XSLT), the following processing instruction is required in the XML:

```
<?xml-stylesheet type="text/xsl" href="myTransform.xslt"?>
```

Client-side XSLT is supported by many web browsers, but not Opera before version 9.0. An alternative,

rather than being dependent on the end-user's browser capabilities, is to use XSL to convert XML into a displayable format *on the server*. The end-user is not aware of what has gone on 'behind the scenes'; all they see is well-formatted, displayable data.

See the XSLT article for an example of server-side XSLT in action.

XML extensions

- **XPath** makes it possible to refer to individual parts of an XML document. This provides random access to XML data for other technologies, including XSLT, XSL-FO, XQuery etc. XPath expressions can refer to all or part of the text, data and values in XML elements, attributes, processing instructions, comments etc. They can also access the names of elements and attributes. XPaths can be used in both valid and well-formed XML, with and without defined namespaces.
- **XInclude** defines the ability for XML files to include all or part of an external file. When processing is complete, the final XML infoset has no XInclude elements, but instead has copied the documents or parts thereof into the final infoset. It uses XPath to refer to a portion of the document for partial inclusions.
- **XQuery** is to XML what SQL and PL/SQL are to relational databases: ways to access, manipulate and return XML.
- **XML Namespaces** enable the same document to contain XML elements and attributes taken from different vocabularies, without any naming collisions occurring.
- **XML Signature** defines the syntax and processing rules for creating digital signatures on XML content.
- **XML Encryption** defines the syntax and processing rules for encrypting XML content.
- **XPointer** is a system for addressing components of XML-based internet media.

XML files may be served with a variety of Media types. RFC 3023 (http://tools.ietf.org/html/rfc3023) defines the types "application/xml" and "text/xml", which say only that the data is in XML, and nothing about its semantics. The use of "text/xml" has been criticized as a potential source of encoding problems but is now in the process of being deprecated.[4] RFC 3023 (http://tools.ietf.org/html/rfc3023) also recommends that XML-based languages be given media types beginning in "application/" and ending in "+xml"; for example "application/atom+xml" for Atom. This page discusses further XML and MIME.

Processing XML files

SAX and DOM are object oriented programming APIs widely used to process XML data. The first XML parsers exposed the contents of XML documents to applications as SAX events or DOM objects.

SAX is a lexical, event-driven interface in which a document is read serially and its contents are reported as "callbacks" to various methods on a handler object of the user's design. SAX is fast and efficient to implement, but difficult to use for extracting information at random from the XML, since it tends to burden the application author with keeping track of what part of the document is being processed. It is better suited to situations in which certain types of information are always handled the same way, no matter where they occur in the document.

DOM is an interface-oriented API that allows for navigation of the entire document as if it were a tree of "Node" objects representing the document's contents. A DOM document can be created by a parser, or can be generated manually by users (with limitations). Data types in DOM Nodes are abstract;

implementations provide their own programming language-specific bindings. DOM implementations tend to be memory intensive, as they generally require the entire document to be loaded into memory and constructed as a tree of objects before access is allowed.

A form of XML access that has become increasingly popular in recent years is push parsing,[5][6] which treats the document as if it were a series of items which are being read in sequence. This allows for writing of recursive-descent parsers in which the structure of the code performing the parsing mirrors the structure of the XML being parsed, and intermediate parsed results can be used and accessed as local variables within the methods performing the parsing, or passed down (as method parameters) into lower-level methods, or returned (as method return values) to higher-level methods. For instance, in the Java programming language, the StAX framework can be used to create what is essentially an 'iterator' which sequentially visits the various elements, attributes, and data in an XML document. Code which uses this 'iterator' can test the current item (to tell, for example, whether it is a start or end element, or text), and inspect its attributes (local name, namespace, values of XML attributes, value of text, etc.), and can also request that the iterator be moved to the 'next' item. The code can thus extract information from the document as it traverses it. One significant advantage of push-parsing methods is that they typically are much more speed- and memory-efficient than SAX and DOM styles of parsing XML. Another advantage is that the recursive-descent approach tends to lend itself easily to keeping data as typed local variables in the code doing the parsing, while SAX, for instance, typically requires a parser to manually maintain intermediate data within a stack of elements which are parent elements of the element being parsed. This tends to mean that push-parsing code is often much more straightforward to understand and maintain than SAX parsing code. Some potential disadvantages of push parsing are that it is a newer approach which is not as well known among XML programmers (although it is by far the most common method used for writing compilers and interpreters for languages other than XML), and that most existing push parsers cannot yet perform advanced processing such as XML schema validation as they parse a document.

Another form of XML Processing API is data binding, where XML data is made available as a custom, strongly typed programming language data structure, in contrast to the interface-oriented DOM. Example data binding systems are the Java Architecture for XML Binding (JAXB)[7] and the Strathclyde Novel Architecture for Querying XML (SNAQue).[8]

A filter in the Extensible Stylesheet Language (XSL) family can transform an XML file for displaying or printing.

- **XSL-FO** is a declarative, XML-based page layout language. An XSL-FO processor can be used to convert an XSL-FO document into another non-XML format, such as PDF.
- **XSLT** is a declarative, XML-based document transformation language. An XSLT processor can use an XSLT *stylesheet* as a guide for the conversion of the data tree represented by one XML document into another tree that can then be serialized as XML, HTML, plain text, or any other format supported by the processor.
- **XQuery** is a W3C language for querying, constructing and transforming XML data.
- **XPath** is a DOM-like node tree data model and path expression language for selecting data within XML documents. XSL-FO, XSLT and XQuery all make use of XPath. XPath also includes a useful function library.

The native file format of OpenOffice.org, AbiWord, and Apple's iWork applications is XML. Some parts of Microsoft Office 2007 is also be able to edit XML files with a user-supplied schema (but not a DTD), and Microsoft has released a file format compatibility kit for Office 2003 that allows previous

versions of Office to save in the new XML based format. There are dozens of other XML editors available.

History

The versatility of SGML for dynamic information display was understood by early digital media publishers in the late 1980s prior to the rise of the Internet.[9][10] By the mid-1990s some practitioners of SGML had gained experience with the then-new World Wide Web, and believed that SGML offered solutions to some of the problems the Web was likely to face as it grew. Dan Connolly added SGML to the list of W3C's activities when he joined the staff in 1995; work began in mid-1996 when Jon Bosak developed a charter and recruited collaborators. Bosak was well connected in the small community of people who had experience both in SGML and the Web. He received support in his efforts from Microsoft.

XML was compiled by a working group of eleven members,[11] supported by an (approximately) 150-member Interest Group. Technical debate took place on the Interest Group mailing list and issues were resolved by consensus or, when that failed, majority vote of the Working Group. The decision record was compiled by Michael Sperberg-McQueen on December 4th 1997.[12] James Clark served as Technical Lead of the Working Group, notably contributing the empty-element "<empty/>" syntax and the name "XML". Other names that had been put forward for consideration included "MAGMA" (Minimal Architecture for Generalized Markup Applications), "SLIM" (Structured Language for Internet Markup) and "MGML" (Minimal Generalized Markup Language). The co-editors of the specification were originally Tim Bray and Michael Sperberg-McQueen. Halfway through the project Bray accepted a consulting engagement with Netscape, provoking vociferous protests from Microsoft. Bray was temporarily asked to resign the editorship. This led to intense dispute in the Working Group, eventually solved by the appointment of Microsoft's Jean Paoli as a third co-editor.

The XML Working Group never met face-to-face; the design was accomplished using a combination of email and weekly teleconferences. The major design decisions were reached in twenty weeks of intense work between July and November of 1996, when the first Working Draft of an XML specification was published.[13] Further design work continued through 1997, and XML 1.0 became a W3C Recommendation on February 10, 1998.

XML 1.0 achieved the Working Group's goals of Internet usability, general-purpose usability, SGML compatibility, facilitation of easy development of processing software, minimization of optional features, legibility, formality, conciseness, and ease of authoring. Like its antecedent SGML, XML allows for some redundant syntactic constructs and includes repetition of element identifiers. In these respects, terseness was not considered essential in its structure.

Sources

XML is a profile of an ISO standard SGML, and most of XML comes from SGML unchanged. From SGML comes the separation of logical and physical structures (elements and entities), the availability of grammar-based validation (DTDs), the separation of data and metadata (elements and attributes), mixed content, the separation of processing from representation (processing instructions), and the default angle-bracket syntax. Removed were the SGML Declaration (XML has a fixed delimiter set and adopts Unicode as the document character set).

Other sources of technology for XML were the Text Encoding Initiative (TEI), which used a simpler syntax than full SGML; HTML, in which elements were synchronous with their resource, the separation of document character set from resource encoding, the xml:lang attribute, and the HTTP notion that metadata accompanied the resource rather than being needed at the declaration of a link; and the Extended Reference Concrete Syntax (ERCS), from which XML 1.0's naming rules were taken, and which had introduced hexadecimal numeric character references and the concept of references to make available all Unicode characters.

Ideas that developed during discussion which were novel in XML, were the algorithm for encoding detection and the encoding header, the processing instruction target, the xml:space attribute, and the new close delimiter for empty element tags.

Versions

There are two current versions of XML. The first, *XML 1.0*, was initially defined in 1998. It has undergone minor revisions since then, without being given a new version number, and is currently in its fourth edition, as published on August 16, 2006. It is widely implemented and still recommended for general use. The second, *XML 1.1*, was initially published on February 4, 2004, the same day as XML 1.0 Third Edition, and is currently in its second edition, as published on August 16, 2006. It contains features — some contentious — that are intended to make XML easier to use in certain cases[14] - mainly enabling the use of line-ending characters used on EBCDIC platforms, and the use of scripts and characters absent from Unicode 2.0. XML 1.1 is not very widely implemented and is recommended for use only by those who need its unique features.

XML 1.0 and XML 1.1 differ in the requirements of characters used for element and attribute names: XML 1.0 only allows characters which are defined in Unicode 2.0, which includes most world scripts, but excludes those which were added in later Unicode versions. Among the excluded scripts are Mongolian, Cambodian, Amharic, Burmese, and others.

Almost any Unicode character can be used in the character data and attribute values of an XML 1.1 document, even if the character is not defined, aside from having a code point, in the current version of Unicode. The approach in XML 1.1 is that only certain characters are forbidden, and everything else is allowed, whereas in XML 1.0, only certain characters are explicitly allowed, thus XML 1.0 cannot accommodate the addition of characters in future versions of Unicode.

In character data and attribute values, XML 1.1 allows the use of more control characters than XML 1.0, but, for "robustness", most of the control characters introduced in XML 1.1 must be expressed as numeric character references. Among the supported control characters in XML 1.1 are two line break codes that must be treated as whitespace. Whitespace characters are the only control codes that can be written directly.

There are also discussions on an XML 2.0, although it remains to be seen if such will ever come about. XML-SW (http://www.textuality.com/xml/xmlSW.html) (SW for skunk works), written by one of the original developers of XML, contains some proposals for what an XML 2.0 might look like: elimination of DTDs from syntax, integration of namespaces, XML Base and XML Information Set (*infoset*) into the base standard.

The World Wide Web Consortium also has an XML Binary Characterization Working Group doing

preliminary research into use cases and properties for a binary encoding of the XML infoset. The working group is not chartered to produce any official standards. Since XML is by definition text-based, ITU-T and ISO are using the name *Fast Infoset (http://asn1.elibel.tm.fr/xml/finf.htm)* for their own binary infoset to avoid confusion (see ITU-T Rec. X.891 | ISO/IEC 24824-1).

Patent claims

In October 2005 the small company Scientigo publicly asserted that two of its patents, U.S. Patent 5,842,213 (http://patft.uspto.gov/netacgi/nph-Parser?patentnumber=5842213) and U.S. Patent 6,393,426 (http://patft.uspto.gov/netacgi/nph-Parser?patentnumber=6393426), apply to the use of XML. The patents cover the "modeling, storage and transfer [of data] in a particular *non-hierarchical*, non-integrated neutral form", according to their applications, which were filed in 1997 and 1999. Scientigo CEO Doyal Bryant expressed a desire to "monetize" the patents but stated that the company was "not interested in having us against the world." He said that Scientigo was discussing the patents with several large corporations.[15]

XML users and independent experts responded to Scientigo's claims with widespread skepticism and criticism. Some derided the company as a patent troll. Tim Bray described any claims that the patents covered XML as "ridiculous on the face of it".[16]

Because there exists a large amount of prior art relating to XML, some legal experts believed it would be difficult for Scientigo to enforce its patents through litigation.

Critique of XML

Commentators have offered various critiques of XML, suggesting circumstances where XML provides both advantages and potential disadvantages.[17]

Advantages of XML

- It is a simultaneously human- and machine-readable format;[18]

- It supports Unicode, allowing almost any information in any written human language to be communicated;
- It can represent the most general computer science data structures: records, lists and trees;
- Its self-documenting format describes structure and field names as well as specific values;
- The strict syntax and parsing requirements make the necessary parsing algorithms extremely simple, efficient, and consistent.

- XML is heavily used as a format for document storage and processing, both online and offline,
- It is based on international standards;
- It allows validation using schema languages such as XSD and Schematron, which makes effective unit-testing, firewalls, acceptance testing, contractual specification and software construction easier;
- The hierarchical structure is suitable for most (but not all) types of documents;
- It manifests as plain text files, which are less restrictive than other proprietary document formats;
- It is platform-independent, thus relatively immune to changes in technology;
- Forward and backward compatibility are relatively easy to maintain despite changes in DTD or

Schema;
- Its predecessor, SGML, has been in use since 1986, so there is extensive experience and software available.

Disadvantages of XML

- XML syntax is redundant or large relative to binary representations of similar data.[19]
- The redundancy may affect application efficiency through higher storage, transmission and processing costs.[20][21]
- XML syntax is too verbose relative to other alternative 'text-based' data transmission formats.[22][23]
- No intrinsic data type support: XML provides no specific notion of "integer", "string", "boolean", "date", and so on.[24][25]
- The hierarchical model for representation is limited in comparison to the relational model or an object oriented graph.[26][27]
- Expressing overlapping (non-hierarchical) node relationships requires extra effort.[28]
- XML namespaces are problematic to use and namespace support can be difficult to correctly implement in an XML parser.[29]
- XML is commonly depicted as "self-documenting" but this depiction ignores critical ambiguities.[30][31]

See also

- Ajax_(programming)
- ASN.1
- asXML
- CDATA section, the mechanism for including non-markup text in XML
- Comparison of layout engines (XML)
- DITA
- DocBook
- ebXML
- Binary XML
- Extensible Binary Meta Language
- Extensible Metadata Platform (XMP), used in graphics applications
- JSON
- List of general purpose markup languages
- List of XML markup languages
- S-expression
- SAML
- Serialization
- Single source publishing
- Structured data interchange
- Universal Business Language
- WBXML
- XML Data Binding
- XML editor
- XML query language
- XML schema languages:
  - DTD,
  - RELAX NG,
  - Schematron,
  - DSDL
  - XML Forms Architecture
- XML Certification Program
- XRI, XDI
- YAML
- W3C XML standards:
  - XForms
  - XHTML
  - XLink
  - XML Base
  - XML Encryption
  - XML-infoset
  - DOM (the XML processing *reference model*).
  - XQuery
  - XML Schema
  - XML Signature
  - XPath
  - XPointer
  - XML Protocol: XMLP and SOAP.

- XML Information Set
- XML processing APIs:
  - DOM,
  - SAX,
  - StAX,
  - E4X
  - VTD-XML
- WSDL, Web service
- XSL and XSLT

Notes and references

1. ^ More precisely, XML is a general-purpose *specification* for creating custom markup languages.
2. ^ Bray, Tim; Jean Paoli, C. M. Sperberg-McQueen, Eve Maler, François Yergeau (September 2006). Extensible Markup Language (XML) 1.0 (Fourth Edition) - Origin and Goals (http://www.w3.org/TR/2006/REC-xml-20060816/#sec-origin-goals). World Wide Web Consortium. Retrieved on October 29, 2006.
3. ^ XHTML is an attempt to simplify and improve the consistency of HTML, which was based on SGML.
4. ^ http://lists.xml.org/archives/xml-dev/200407/msg00208.html
5. ^ Fitzgerald, Michael (2004). *Learning XSLT.* Sebastopol, CA: O'Reilly Media, p.243. ISBN 0-596-00327-7.
6. ^ Push, Pull, Next! (http://www.xml.com/pub/a/2005/07/06/tr.html) by Bob DuCharme, at XML.com
7. ^ http://java.sun.com/xml/jaxb/
8. ^ http://www.cis.strath.ac.uk/research/snaque/
9. ^ Bray, Tim (February 2005). A conversation with Tim Bray: Searching for ways to tame the world's vast stores of information (http://www.acmqueue.com/modules.php?name=Content&pa=showpage&pid=282). Association for Computing Machinery's "Queue site". Retrieved on April 16, 2006.
10. ^ (1988) "Publishers, multimedia, and interactivity", *{{{title}}}*. Cobb Group. ISBN 1-55615-124-1.
11. ^ The working group was originally called the "Editorial Review Board." The original members and seven who were added before the first edition was complete, are listed at the end of the first edition of the XML Recommendation, at http://www.w3.org/TR/1998/REC-xml-19980210.
12. ^ Reports From the W3C SGML ERB to the SGML WG And from the W3C XML ERB to the XML SIG (http://www.w3.org/XML/9712-reports.html)
13. ^ http://www.w3.org/TR/WD-xml-961114.html
14. ^ Extensible Markup Language (XML) 1.1 (Second Edition) - Rationale and list of changes for XML 1.1 (http://www.w3.org/TR/xml11/#sec-xml11). W3C. Retrieved on 2006-12-21.
15. ^ http://news.com.com/Small+company+makes+big+claims+on+XML+patents/2100-1014_3-5905949.html
16. ^ http://blogs.zdnet.com/BTL/?p=2052
17. ^ (See e.g., XML-QL Proposal discussing XML benefits (http://www.w3.org/TR/NOTE-xml-ql/), When to use XML (http://www.25hoursaday.com/weblog/PermaLink.aspx?guid=dada27bf-2af0-400d-94c9-5575546f5664), "XML Sucks" on c2.com (http://c2.com/cgi/wiki?XmlSucks), Daring to Do Less with XML (http://www.xml.com/pub/a/2001/05/02/champion.html))
18. ^ See XML in ten points (http://www.w3.org/XML/1999/XML-in-10-points). Some documents, however may be technically *readable* but too cluttered to be easily legible or comprehensible. This critique is especially common for XML files with extensive use of XML Namespaces, character entity references and other complications.
19. ^ Rusty, Elliotte (2002). *Processing XML with Java(tm): a guide to SAX, DOM, JDOM, JAXP, and TrAX.* Addison-Wesley. 0201771861. XML documents are too verbose compared with binary equivalents.
20. ^ Rusty, Elliotte (2002). *Xml in a Nutshell: A Desktop Quick Reference.* O'Reilly. 0596002920. XML documents are very verbose and searching is inefficient for high-performance largescale database applications.
21. ^ However, the Binary XML effort strives to alleviate these problems by using a binary representation for the XML document. For example, the Java reference implementation of the Fast Infoset standard parsing speed is better by a factor 10 compared to Java Xerces, and by a factor 4 compared to the Piccolo driver (http://piccolo.sourceforge.net/), one of the fastest Java-based XML parser [1] (https://fi.dev.java.net/reports/parsing/report.html).
22. ^ Bierman, Gavin (2005). *Database Programming Languages: 10th international symposium, DBPL 2005*

Trondheim, Norway. Springer. 3540309519. XML syntax is too verbose for human readers in for certain applications. Proposes a dual syntax for human readability.
23. ^ Although many purportedly "less verbose" text formats actually cite XML as both inspiration and prior art. See e.g., http://yaml.org/spec/current.html, http://innig.net/software/sweetxml/index.html, http://www.json.org/xml.html.
24. ^ For example, there is no provision for mandating that "3.14159" be a floating-point number rather than a seven-character string. Such conversions require processing. However, XML schema descriptions of XML documents add data-type support to XML processing.
25. ^ Blanken, Henk M. (2003). *Intelligent Search on Xml Data: Applications, Languages, Models, Implementations, and Benchmarks*. Springer. 3540407685. Proposes a system for recognizing custom-defined data types.
26. ^ A hierachical model only gives a fixed, monolithic view of the tree structure. For example, either actors under movies, or movies under actors, but not both.
27. ^ Lim, Ee-Peng (2002). *Digital Libraries: People, Knowledge, and Technology*. Springer. 3540002618. Discusses some of the limitation with fixed hierarchy. Proceedings of the 5th International Conference on Asian Digitial Libraries, ICADL 2002, held in Singapore in December 2002.
28. ^ Searle, Leroy F. (2004). *Voice, text, hypertext: emerging practices in textual studies*. University of Washington Press. 0295983051. Proposes an alternative system for encoding overlapping elements.
29. ^ (See e.g., http://www-128.ibm.com/developerworks/library/x-abolns.html)
30. ^ The Myth of Self-Describing XML (http://www.oceaninformatics.biz/publications/e2.pdf). Retrieved on 2007-05-12.
31. ^ (See e.g., Use-mention distinction, Naming collision, Polysemy)

External links

Specifications

- W3C XML homepage (http://www.w3.org/XML/)
- The XML 1.0 specification (http://www.w3.org/TR/REC-xml)
- The XML 1.1 specification (http://www.w3.org/TR/xml11)

Parsers

- Expat free stream-oriented XML 1.0 parser library, written in C.
- Libxml2 free XML C parser and toolkit.
- RomXML (http://www.allegrosoft.com/romxml.html) Embedded XML commercial toolkit written in ANSI-C.
- XML resources (http://dmoz.org/Computers/Data_Formats/Markup_Languages/XML) at the Open Directory Project

Sources

- Introduction to Generalized Markup (http://www.sgmlsource.com/history/AnnexA.htm) by Charles Goldfarb
- Annex A of ISO 8879:1986 (SGML)
- The Multilingual WWW (http://www.mind-to-mind.com/library/papers/multilingual/multilingual-www.html) by Gavin Nicol
- Retrospective on Extended Reference Concrete Syntax (http://xml.ascc.net/en/utf-8/ercsretro.html) by Rick Jelliffe
- XML, Java and the Future of the Web (http://www.xml.com/pub/a/w3j/s3.bosak.html) by Jon Bosak

Retrospectives

- Thinking XML: The XML decade (http://www-128.ibm.com/developerworks/library/x-think38.html) by Uche Ogbuji
- XML: Ten year anniversary (http://drmacros-xml-rants.blogspot.com/2006/11/xml-ten-year-aniversary.html) by Elliot Kimber
- Closing Keynote, XML 2006 (http://2006.xmlconference.org/proceedings/162/presentation.html) by Jon Bosak
- Five years later, XML... (http://www.oreillynet.com/xml/blog/2003/02/five_years_later_xml.html) by Simon St. Laurent

Papers

- Lawrence A. Cunningham (2005). "Language, Deals and Standards: The Future of XML Contracts". *Washington University Law Review*. SSRN 900616 (http://ssrn.com/abstract=900616).

Retrieved from "http://en.wikipedia.org/wiki/XML"

Categories: Wikipedia articles needing clarification | Articles with unsourced statements since February 2007 | All articles with unsourced statements | XML | Markup languages | W3C standards | Technical communication | Computer file formats | Open formats | Data modeling languages

- This page was last modified 09:44, 21 June 2007.
- All text is available under the terms of the GNU Free Documentation License. (See **Copyrights** for details.) Wikipedia® is a registered trademark of the Wikimedia Foundation, Inc., a US-registered 501(c)(3) tax-deductible nonprofit charity.

about XML . XML Activity Statement

Extensible Markup Language (XML)

1. Introduction
2. Working Groups
3. Other Resources
4. Contact

Nearby: XML Specifications and Translations of them.

Introduction

Extensible Markup Language (XML) is a simple, very flexible text format derived from SGML (ISO 8879). Originally designed to meet the challenges of large-scale electronic publishing, XML is also playing an increasingly important role in the exchange of a wide variety of data on the Web and elsewhere.

This page describes the work being done at W3C within the XML Activity, and how it is structured. Work at W3C takes place in *Working Groups*. The Working Groups within the XML Activity are listed below, together with links to their individual web pages.

You can find and download formal technical specifications here, because we publish them. This is ***not*** a place to find tutorials, products, courses, books or other XML-related information. There are some links below that may help you find such resources.

You will find links to W3C Recommendations, Proposed Recommendations, Working Drafts, conformance test suites and other documents on the pages for each Working Group. Each document also contains email addresses you can use to send comments or questions, for example if you have been writing software to implement them and have found problems or errors.

Please do ***not*** send us email asking us to help you learn a language or specification; there are plenty of resources online, and the people editing and developing the specifications are very busy. We ***are*** interested in technical comments and errata.

If your organization would like to join the W3C, or if you would like to participate formally in a working group (and have the necessary resources to attend meetings), you can read more about the Consortium.

Working Groups

There is more detail about each of these Working Groups in the Activity Statement and also on the individual Working Group public web pages.

Most Working Groups have both a public web page and another more private one that is only accessible to W3C Members. The private page has telephone numbers, schedules for meetings and conference calls, links to internal editing drafts, and other administrative information.

XML Coordination Group

The membership of this group is the Chairs of the individual Working Groups. Its role is to provide a forum for coordination between the Working Groups of the XML Activity, and between the XML Activity and other parts of W3C, and between the XML Activity and other organizations.

This group does not produce specifications, so does not have a public page of its own. You can read the XML CG Charter, and there is more information about the XML CG in the Activity Statement. There is is also a member-only page.

XML Core Working Group

The mission of the XML Core Working Group is to develop and maintain the specifications for XML itself and closely related specifications such as Namespaces in XML, the XML Information Set, and XInclude.

You can read the XML Core Working Group Public Page and the XML Core Working Group Charter, and there is also a member-only page.

XSL Working Group

The XSL Working Group is responsible for the Extensible Stylesheet Language (XSL), including both XSL Transformations (XSLT) and XSL Formatting Objects (XSL/FO). They moved into the W3C Architecture Domain at the start of 2003.

You can read the XSL Working Group Public Page which links to their Charter and to their member-only page.

The Efficient XML Interchange Working Group

The Efficient XML Interchange Working Group is responsible for developing ways to exchange XML documents in ways that are as efficient as is practical without compromising the interoperability of XML itself. It also continues the work of the XML Binary Characterization Working Group. This Working Group is ***not*** about producing a closed, proprietary or obfuscated "binary XML "—The W3C is all about increasing interoperability!

You can read the Efficient XML Interchange Working Group Public Page and their Charter; there is also a member-only page for technical discussion and administration.

XML Binary Characterization Working Group

In September of 2003 the W3C held a public Workshop to discuss whether W3C should develop a binary interchange format for people needing greater efficiency than was claimed possible using text. The conclusion of the Workshop was that it was not clear whether the benefits of such a format could outweigh the costs, but that the issues involved needed further study. The XML Binary Characterization Working Group was created to investigate this area, and to enable precise measurements to be made of the benefits (if any) of a binary interchange format over the existing methods of textual interchange. The Working Group was explicitly ***not*** chartered to do any technical work on developing or choosing a single format: that would be the work of a future Working Group, if and only if the XML Binary Characterization Working Group can demonstrate that such efforts would be worth while.

You can read the XML Binary Characterization Working Group Public Page which links to their charter and to their Member-only page.

XML Processing Model Working Group

The XML Processing Model Working Group is working on defining a scripting language for XML: that is, a way to specify what operations should be performed on an XML document and in what order.

You can read the XML Processing Model Working Group Public Page and their Charter; there is also a member-only page for administration purposes.

XML Linking Working Group

The charter of the XML Linking Working Group has expired, and the group is not currently active. When still active, it was working on hypertext links for XML. This includes the XML Linking Language (XLink) and the XML Pointer Language (XPointer).

You can read the deprecated XML Linking Working Group Public Page. There is also a member-only page.

XML Query Working Group

The XML Query Working Group is working on the XML Query Language, a way to provide flexible query facilities to extract data from real and virtual XML documents on the Web. This includes publication of XQuery and also XPath, in conjunction with the XSL Working Group (part of the Style Activity).

You can read the XML Query Working Group Public Page and the XML Query Working Group Charter, and there is also a member-only page.

XML Schema Working Group

W3C XML Schemas provide mechanisms to define and describe the structure, content, and to some extent semantics of XML documents.

You can read the XML Schema Group Public Page and the XML Schema Working Group Charter, and there is also a member-only page.

Service Modeling Language Working Group

SML defines extensions to the W3C XML Schema language by adding support for inter-document references and user-defined constraints.

You can read the SML Working Group Public Page and the SML Working Group Charter.

Other Resources

There are so many resources related to XML that we can't possibly list them all here. This is a ***good*** thing, because it means XML is a success! In addition to a history of the development of XML at W3C, there is an extensive index at the Cover Pages, maintained by Robin Cover. The individual Working Group public web pages may have links to specific resources. There are Usenet newsgroups (e.g. comp.text.xml) and public mailing lists (e.g. xml-dev).

You could also try a search engine such as Google for:

- XML conferences
- books
- training courses
- online tutorials
- bibliographies
- parsers (both proprietary and open source)
- magazines
- and even movies

This page is also available in Polish (polski)

Contact

Liam Quin, XML Activity Lead

Last modified $Date: 2007/05/08 17:40:49 $

Copyright © 1996-2003 W3C® (MIT, ERCIM, Keio), All Rights Reserved. W3C liability, trademark, document use and software licensing rules apply. Your interactions with this site are in accordance with our public and Member privacy statements.

Note

The XML specification, and other information specific to the XML Core Working Group, has moved to the XML Core Working Group Public Page.

There is also a separate page for Translations.

There is a separate page documenting the xml-spec DTD used for many of our specifications.